CS 134: Networks

Prof. Yaron Singer Spring 2017

Material for this handout is drawn heavily from "Structure and Dynamics of Information in Networks".

1 s-t Cuts for Clustering

Consider the problem of community detection given $G = (V, E)$. We might want to detect a community in V such that nodes in V have many links with other nodes in V, but relatively few links with nodes outside of V. If we want to detect multiple communities—say, two communities—within a graph, we might want to partition the nodes in V into a cut (a pair of disjoint sets) that minimizes the number of edges between the cut. We use the following notation to begin formalizing this intuition.

Definition. *Let $G = (V, E)$ be a graph.*

1. *$e(S,T) = E \cap (S \times T)$ denotes the set of edges with exactly one endpoint in S and one in T. $e(S) = e(S,S)$ is the set of edges with both endpoints in S.*
2. *We use $d_S(v) = |e(\{v\}, S)|$ to denote the degree of v within the set S, i.e., number of edges between v and nodes in S.*
3. *The edge density of a node-set S is defined as $\frac{|e(S)|}{|S|}$.*

Now, consider a graph G. The simplest definition of a community is a dense subgraph, i.e., a subgraph with large edge density in the sense of $e(S)$. If we want to find the *best* community, that would be the subgraph with the largest density.

We can begin by formulating finding the "densest" subgraph as a decision problem: that is, given some constant α, is there a subgraph S with $\frac{|e(S)|}{|S|} \geq \alpha$? Or, equivalently, is there some S such that $|e(S)| - \alpha|S| \geq 0$?

We can also re-write $e(S)$. Note that internal edges of S contribute 2 to the total degree in S, and each edge leaving contributes 1. Therefore, $2|e(S)| = \frac{1}{2}(\sum_{v \in S} d(v) - |e(S, S^{\complement})|)$. Substituting this expression into the inequality above, we have

$$|e(S)| = \sum_{v \in S} d(v) - |e(S, S^{\complement})|$$

Therefore we can rewrite the inequality as

$$\sum_{v \in S} d(v) - |e(S, S^{\complement})| - 2\alpha|S| \geq 0$$

We can rewrite above as

$$\sum_{v \in V} d(v) - (\sum_{v \in S^{\complement}} d(v) + |e(S, S^{\complement})| + 2\alpha|S|) \geq 0$$

Given that the first term, $\sum_{v \in V} d(v)$, is equal to $2|E|$ and is therefore constant, this constraint is satisfiable if and only if it is satisfied by the set S that minimizes $\beta(s) = (\sum_{v \in S^\complement} d(v) + |e(S, S^\complement)| + 2\alpha|S|)$.

To check our satisfiability constraint, we can rewrite this problem as a mincut problem. Consider the grpah G' with $V' = V \cup \{s, t\}$ where s is connected to all vertices v with an edge of capacity $d(v)$, and t is connected to all vertices in V with an edge of capacity 2α. The cost of the cut $(S + s, S^\complement + t)$ in G' is exactly $\beta(s)$. Thus if the minimum $s - t$ cut $(S + s, S^\complement + t)$ of G' satisfies the constraint of being less than or equal to $2|E|$, then S is a set of density of at least α.

However, when we first formulated this problem, we didn't just want to check a satisfiability condition; we wanted to find the maximal α. We could do this by performing binary search over all α. or we could use some efficient max-flow algorithms (no need to worry about this!) to compute $s - t$ cuts for all values of α in one computation. This can be done in $O(n^2 m)$ time.

We might also want to, given a graph G, find the densest subgraph S containing a specific vertex set X (i.e., S maximizes $\frac{|e(S)|}{|S|}$ over all sets $S \supseteq X$). We can solve this problem using the above method simply by giving all edges from s to vertices $v \in X$ a capacity of ∞. This will ensure that all nodes in X are on the s-side of the cut, and the rest of the analysis stays the same.

2 A 1/2-Approximation for Clustering

In the previous section, we found an algorithm that found the densest subgraph in $O(mn^2)$ time. However, in real-world graphs, m and n may be so large that this runtime is prohibitively slow. Are there linear or near-linear time algorithms?

Since, from the previous section, we know that the $s-t$ cut problem is interlinked with the densest subgraph problem, and known ways for computing $s - t$ cuts aren't much faster than $O(mn^2)$, we'll have to resort to an approximation algorithm. The following is a greedy algorithm for getting within a factor of 1/2 for finding dense subgraphs.

ALG 1: A Greedy $\frac{1}{2}$-Approximation Algorithm for finding dense subgraphs
Let $G_n \leftarrow G$
for $k = n$ downto $\|X\| + 1$ **do**
Let $v \notin X$ be the lowest degree node in $G_k \setminus X$.
Let $G_{k-1} \leftarrow G_k \setminus \{v\}$.
Output the densest subgraph among $G_n, \ldots, G_{\|X\|}$.

The above algorithm is a $\frac{1}{2}$-approximation. Let $S \supseteq X$ be the densest subgraph. If our algorithm outputs S, then it is optimal. If not, at some point, we must have deleted a node $v \in S$. Let G_k be the graph right before the first $v \in S$ was incorrectly removed. Because S is optimal, removing v from it would only make it less dense, so

$$\frac{|e(S)|}{|S|} \geq \frac{|e(S - v)|}{|S| - 1} \geq \frac{|e(S)| - d_s(v)}{|S| - 1}$$

Multiplying through with $|S|(|S| - 1)$ and rearranging gives us $d_s(v) \geq \frac{|e(S)|}{|S|}$.

Because G_k is a supergraph of S, the degree of v in G_k must be at least as large as in S, so $d_{G_k}(v) \geq d_S(v) \geq \frac{|e(S)|}{|S|}$. The algorithm chose v (however mistakenly!) because it had the minimum degree, so we know that for each $u \in G_k \setminus X$, we have $d_{G_k}(u) \geq d_{G_k}(v) \geq \frac{|e(S)|}{|S|}$. We then obtain the bound on the density of the graph G_k:

$$\begin{aligned}
\frac{|e(G_k)|}{|G_k|} &\geq \frac{\sum\limits_{u \in S} d_S(u) + \sum\limits_{u \in G_k \setminus S} \frac{e(S)}{|S|}}{2|G_k|} \\
&= \frac{2|e(S)| + |G_k \setminus S| \frac{|e(S)|}{|S|}}{2|G_k|} \\
&\geq \frac{|e(S)|}{|S|} \cdot \frac{|S| + |G_k \setminus S|}{2|G_k|} \\
&= \frac{|e(S)|}{2|S|}
\end{aligned}$$

The graph that the algorithm outputs is certainly no worse than G_k, as G_k was available as a potential option. Hence, the algorithm is a $\frac{1}{2}$-approximation.